FITS Data Format for Calibrated Imaging Data

John Young, Univ. Cambridge

Tom Pauls, NRL

http://www.mrao.cam.ac.uk/~jsy1001/exchange/

Acknowledgements

- Bill Cotton
- David Buscher
- John Monnier
- Dave Mozurkewich
- Tom Armstrong
- Pascal Ballester
- Nick Elias
- Thierry Forveille
- Gilles Duvert
- Andy Boden
- Those at AAS198 round table discussion

Outline

- History
- Why?
- Current status
- Format description
  - Scope
  - FITS Primer
  - File structure
  - Software
  - FAQs
- Proposed changes
- Possible topics for discussion

History

- Idea from June 2000 NSF-sponsored meeting in Socorro
- Since 2001, TP & JSY + community have been working on the format
- Drafts discussed at
  - AAS 198, June 2000
  - IAU WG Meeting, August 2001
- Public pre-release, March 2002
  - document + example code

Why a common data format?

- Share data
  - With collaborators (combine datasets)
  - End-users of facility arrays
- Share software
  - Save effort
  - Get better software

Current status

- Pre-release!
- Specification document
- Example C code (cfitsio)
- John Monnier's IDL reader/writer code

See

`http://www.mrao.cam.ac.uk/~jsy1001/exchange/`

Preliminary Draft

A Data Exchange Standard for Optical/IR Interferometry

Prepared by NPOI and COAST

First Version: 10 January 2001

Last Update: 25 April 2002

1 Introduction

Considering the number of optical and infrared interferometers in operation or under construction it seems imperative that a standard be established which will allow the exchange of data among various groups. As a first step, we should try to agree on a standard for exchanging *calibrated* data. By first concentrating on calibrated data we only need those instrumental parameters which are required to characterize the data for subsequent analysis. Our ultimate goal is to cast the standard as a FITS file format definition, but first we should agree on what should be included in the standard. The initial document is designed to standardize the exchange of imaging data, but does not preclude the possibility of including other types of data in the future (e.g. astrometry data).

Jaffe and Cotton have been working on a FITS file format for use with the VLTI backends. While the Jaffe and Cotton document contains elements to handle the storage of raw data, many of their elements are also needed for general data exchange. In this document we summarize the information we think is needed for data exchange.

Document Conventions

In what follows we have tried to follow the FITS binary table usage of keywords and column headings. The keywords can be considered as scalars, while the columns can be simply an array, or an array of pointers to other arrays. Allowed data types are: **I** = integer (16-bit), **A** = character, **E** = real (32-bit), **D** = double (64-bit), **L** = logical, **M** = complex (2 64-bit). The number in parentheses is the dimensionality of the entry.

Mandatory keywords describing the structure of the FITS binary tables have been omitted (see "Definition of The Flexible Image Transport System"). The reference also describes various extensions to binary tables that are not part of the FITS standard. None of these are currently used in this format.

The revision numbers of all tables are currently zero. After a suitable amount of discussion and consequent changes, the table definitions will be "frozen", and assigned revision numbers of one. Further changes will require increments in the appropriate revision numbers.

What it is/isn't

- Aimed at imaging applications
- Calibrated (and averaged) data only
- Not astrometry (yet?)
- Not space interferometry
- Not an archive format
  - Minimalist

But FITS is flexible – you can add extra tables

FITS primer

- Originally for images only. Now much more flexible
- Sequence of Header + Data units (HDUs):
  - ASCII header: **keywords** & values
  - Data, one of:
    - Image
    - ASCII Table (not in 1st HDU)
    - Binary Table (not in 1st HDU)
- We only use **binary tables**:
  - Columns (with **headings**)
  - Values in columns may be scalars or vectors
  - Many data types (integer, real, complex)
  - Table structure described by mandatory keywords in header
- Relational database

Example bintable header

```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / 8-bit bytes
NAXIS   =                    2 / 2-dimensional binary table
NAXIS1  =                   46 / width of table in bytes
NAXIS2  =                    4 / number of rows in table
PCOUNT  =                    0 / size of special data area
GCOUNT  =                    1 / one data group (required keyword)
TFIELDS =                    5 / number of fields in each row
TTYPE1  = 'TEL_NAME'           / label for field   1
TFORM1  = '8A      '           / data format of field: ASCII Character
TTYPE2  = 'STA_NAME'           / label for field   2
TFORM2  = '8A      '           / data format of field: ASCII Character
TTYPE3  = 'INDEX   '           / label for field   3
TFORM3  = 'I       '           / data format of field: 2-byte INTEGER
TTYPE4  = 'DIAMETER'           / label for field   4
TFORM4  = 'E       '           / data format of field: 4-byte REAL
TUNIT4  = 'm       '           / physical unit of field
TTYPE5  = 'STAXYZ  '           / label for field   5
TFORM5  = '3D      '           / data format of field: 8-byte DOUBLE
TUNIT5  = 'm       '           / physical unit of field
EXTNAME = 'OI_ARRAY'           / name of this binary table extension
REVISION=                    0 / Revision number of the table definition
ARRNAM  = 'COAST   '           / Array name
FRAME   = 'GEOCENTRIC'         / Coordinate frame
ARRAYX  =             3920635. / [m] Array centre x coordinate
ARRAYY  =               -2889. / [m] Array centre y coordinate
ARRAYZ  =             5013987. / [m] Array centre z coordinate
DATE-OBS= '2000-10-19'         / Start date of observations
END
```

Example bintable data

Displayed by **fv** (part of ftools):

fv : Binary Table of testdata.fits[1] in /home/jsy1001/src/exchange

File Edit Plot Sort Calc Histogram Statistics Help

	TEL_NAMI	STA_NAMI	INDEX	DIAMETER	STAXYZ
	8A	8A	I	E	3D
				m	m
					expand
1	TEL2	C	1	4.0000E-01	0.0000000000E+00
2	TEL4	W4	2	4.0000E-01	4.4740000000E+00
3	TEL3	E3	3	4.0000E-01	3.6320000000E+00
4	TEL1	N3	4	4.0000E-01	-1.1240000000E+01

Go to: Edit cell:

Defined tables

- Separate binary table for each data type:
  - Squared visibility (OI_VIS2)
  - Complex visibility (OI_VIS)
  - Triple product (OI_T3)
- Accompanying OI_WAVELENGTH table
- Data tables include definitive *u*, *v* coordinates
- Table with minimal target information (OI_TARGET)
- Table giving info needed to reproduce/interpolate *u*, *v* coordinates
  - OI_ARRAY (or alternative)

Undefined tables/features

- Equivalents to OI_ARRAY for aperture masking, orbiting arrays etc.
- Instrument-specific information
- Astrometric information (internal metrology etc.)
- Polarisation state
- Description of u, v plane averaging

Using the example code

Example routines read/write from/to data structures (C structs) in memory. These structures mimic the file structure as far as possible.

Two ways to use the routines:

1. Use routines as is, your code interfaces to the data structures provided.
2. Create equivalent I/O routines that read/write from/to more appropriate data structures.

```
/**
 * Write OI_WAVELENGTH fits binary table
 *
 *   @param fptr   see cfitsio documentation
 *   @param wave   wavelength data struct, see exchange.h
 *   @param status pointer to status variable
 *
 *   @return On error, returns non-zero cfitsio error code, and sets status
 */
int write_oi_wavelength(fitsfile *fptr, oi_wavelength wave, int *status) {

  const char function[] = "write_oi_wavelength";
  const int tfields = 2;
  char *ttype[] = {"EFF_WAVE", "EFF_BAND"};
  char *tform[] = {"E", "E"};
  char *tunit[] = {"m", "m"};
  char extname[] = "OI_WAVELENGTH";
  int revision = 0;

  if (*status) return *status; /* error flag set - do nothing */
  fits_create_tbl(fptr, BINARY_TBL, 0, tfields, ttype, tform, tunit,
                  extname, status);
  if (wave.revision != revision) {
    printf("WARNING! wave.revision != %d on entry to write_oi_wavelength.
Writing revision %d table\n", revision, revision);
  }
  fits_write_key(fptr, TINT, "REVISION", &revision,
                 "Revision number of the table definition", status);
  fits_write_col(fptr, TFLOAT, 1, 1, 1, 1, wave.eff_wave, status);
  fits_write_col(fptr, TFLOAT, 2, 1, 1, 1, wave.eff_band, status);

  if (*status) {
    fprintf(stderr, "CFITSIO error in %s:\n", function);
    fits_report_error(stderr, *status);
  }
  return *status;
}
```

Other software

- Completed
  - IDL reader/writer (John Monnier)
    - OYSTER (USNO)
- Under construction
  - AIPS++ interface (NRL)
  - Model-fitting software (Univ. Cambridge)
  - BSMEM (Univ. Cambridge)

Frequently Asked Questions

- Wouldn't *xxx* be a better name for this keyword/column name? Who chose that stupid name anyway?
- I would like to store this extra piece of information. Can we add a keyword/column for it?
  - Calibrator codes.
  - Description of *u*, *v* averaging.
- Please explain VELTYP and VELDEF.

Proposed changes

Mostly still aimed at imaging

- Keyword/column name changes
- Longer string fields
- Make OI_ARRAY table optional
  - But it contains DATE-OBS
- Explicitly allow extra columns

Suggested topics for discussion

- Proposed changes
- Routes to new applications
  - Astrometry
  - Aperture masking
  - Polarimetry
- Prospects for new software
- When to freeze the specification